# TIME-DEPENDENT SIMULATION OF INCOMPRESSIBLE FLOW IN A TURBOPUMP USING OVERSET GRID APPROACH

Cetin Kiris, and Dochan Kwak
M.S. T27B, NAS Applications Branch
NASA Advanced Supercomputing (NAS) Division
NASA-Ames Research Center, Moffett Field, CA 94035

## ABSTRACT

This paper reports the progress being made towards complete unsteady turbo-pump simulation capability by using overset grid systems. A computational model of a turbo-pump impeller is used as a test case for the performance evaluation of the MPI, hybrid MPI/Open-MP, and MLP versions of the INS3D code. Relative motion of the grid system for rotor-stator interaction was obtained by employing overset grid techniques. Unsteady computations for a turbo-pump, which contains 114 zones with 34.3 Million grid points, are performed on Origin 2000 systems at NASA Ames Research Center. The approach taken for these simulations, and the performance of the parallel versions of the code are presented.

## INTRODUCTION

The motivation of this effort is based on two primary elements. First, the entire turbo pump simulation intends to provide a computationai framework for the design and analysis for the liquid rocket engine fuel supply system. This effort is part of the High Performance Computing and Communications (HPCC) advanced technology application projects. The second objective of this research is to support the design of liquid rocket systems for the space transportation. Since the space launch systems in the near future are likely to rely on liquid rocket engines, increasing the efficiency and reliability of the engine components is an important task. One of the major problems in the liquid rocket engine is to understand the fluid dynamics of fuel and oxidizer flows from the fuel tank to plume. Understanding the flow in the turbo pump through numerical simulation will be of significant value toward finding a better design that is simpler yet more efficient and robust with less manufacturing cost. Until recently, the pump design process was not significantly different from that of decades ago. The current semi-empirical turbomachinary design process does not account for the three-dimensional (3-D) viscous phenomena in the pump flows. Some of these 3-D viscous phenomena include wakes; the boundary layers in the hub, the shroud

and the blades; junction flows; and tip clearance flows. Even though computational fluid dynamics (CFD) applications in turbines have been reported widely in the literature, the applications in entire-pump simulations are quite limited. The objective of this paper is to present, and evaluate a parallel computational procedure that simulates the incompressible flow through the entire turbo pump configuration.

A substantial computational time reduction for these 3D unsteady flow simulations is required to reduce design cycle time of the pumps. Part of this speed up will be due to enhancements in computer hardware platforms. The remaining portion of the speed-up will be contributed by advances in algorithms and by efficient parallel implementations. The following section outlines the initial effort and steps taken in order to reach this speed-up.

## NUMERICAL METHOD

The present computations are performed utilizing the INS3D computer code, which solves the incompressible Navier-Stokes equations for both steady-state and unsteady flows. The numerical solution of the incompressible Navier-Stokes equations requires special attention in order to satisfy the divergence-free constraint on the velocity field because the incompressible formulation does not yield the pressure field explicitly from the equation of state or through the continuity equation. One way to avoid the numerical difficulty originated by the elliptic nature of the problem is to use an artificial compressibility method, developed by Chorin[1]. The artificial compressibility algorithm, which introduces a time-derivative of the pressure term into the continuity equation; the elliptic-parabolic type partial differential equations are transformed into the hyperbolic-parabolic type. A family of flow solvers has been developed [2-3] based on this algorithm. Since the convective terms of the resulting equations are hyperbolic, upwind differencing can be applied to these terms. The current versions, designated as INS3D code, use Roe's flux-difference splitting[4]. The third and fifth-order upwind differencing used here is an implementation of a class of high-accuracy flux-differencing schemes for the compressible flow equations. To obtain time-accurate solutions, the equations are iterated to convergence in pseudo-time for each physical time step until the divergence of the velocity field has been reduced below a specified tolerance value. The total number of sub-iteration required varies depending on the problem, time step size and the artificial compressibility parameter used, and typically ranges from 10 to 30 sub-iterations. The matrix equation is solved iteratively by using a non-factored Gauss-Seidel type line-relaxation scheme[5], which maintains stability and allows a large pseudo-time step to be taken.

Details of the numerical method can be found in Refs. 2-3. The GMRES scheme has also been utilized for the solution of the resulting matrix equation[6]. Computer memory requirement for the flow solver INS3D with line-relaxation is 35 times the number of grid points in words, and with GMRES-ILU(0) scheme is 220 times the number of grid points in words. When a fast converging scheme, such as a

GMRES-ILU(0) solver, was implemented into the artificial compressibility method, previous computations showed that reasonable agreement was obtained between computed results and experimental data. The line-relaxation scheme in artificial compressibility method could be very expensive for time accurate computations and could lead to erroneous solutions if incompressibility is not enforced in each physical time step.

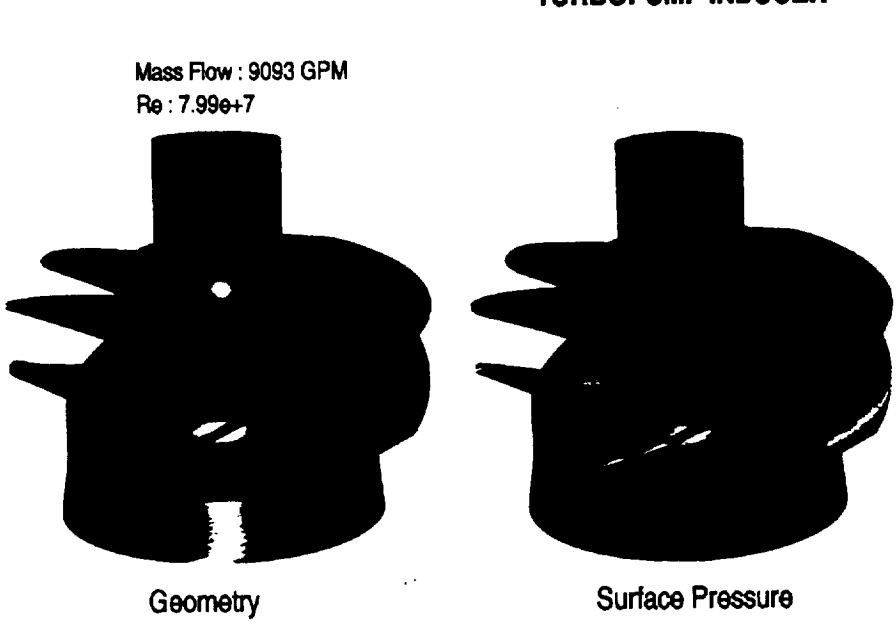


**Figure 1**. Geometry and surface pressure for a pump inducer.

## APPROACH AND COMPUTATIONAL MODELS

The geometry for the liquid oxygen pump has various rotating and stationary components, such as inducer, stators, kicker, and hydraulic turbine, where the flow is extremely unsteady. Figure 1 shows the geometry and computed surface pressure of the inducer from steady-state components analysis. When rotating and stationary parts are included, time-dependent simulations need to be carried out due to unsteady interactions.

To handle the geometric complexity, an overset grid approach is used. The overset structured grid approach to flow simulation has been utilized to solve a variety of problems in aerospace, marine, biomedical and meteorological applications. Flow regimes can range from simple steady flows as that of a commercial aircraft, to unsteady three-dimensional flows with bodies in relative motion as in the case of turbopump configurations. A geometrically complex body is decomposed into a number of simple grid components, as shown in figure 2. The freedom to allow neighboring grids to overlap arbitrarily implies these grids can be created independently from each other and each grid is typically of high quality and nearly orthogonal. Connectivity between neighboring grids is

established by interpolation at the grid outer boundaries. Addition of new components to the system and simulating arbitrary relative motion between multiple bodies are achieved by establishing new connectivity without disturbing the existing grids. Scalability on parallel compute platforms is naturally accomplished by the already decomposed grid system. For certain problems, it is more efficient to gather the grids into groups of approximately equal sizes for parallel processing.

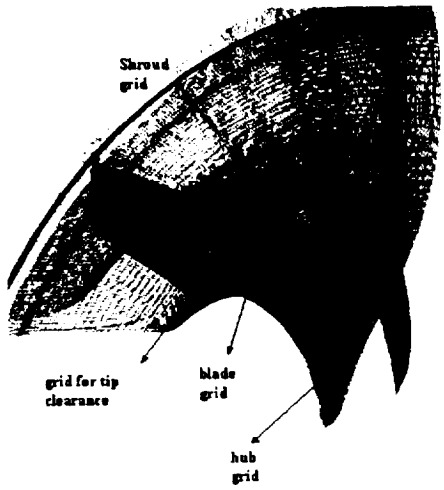


**Figure 2.** Overset grid system for the impeller long blade section with tip clearance

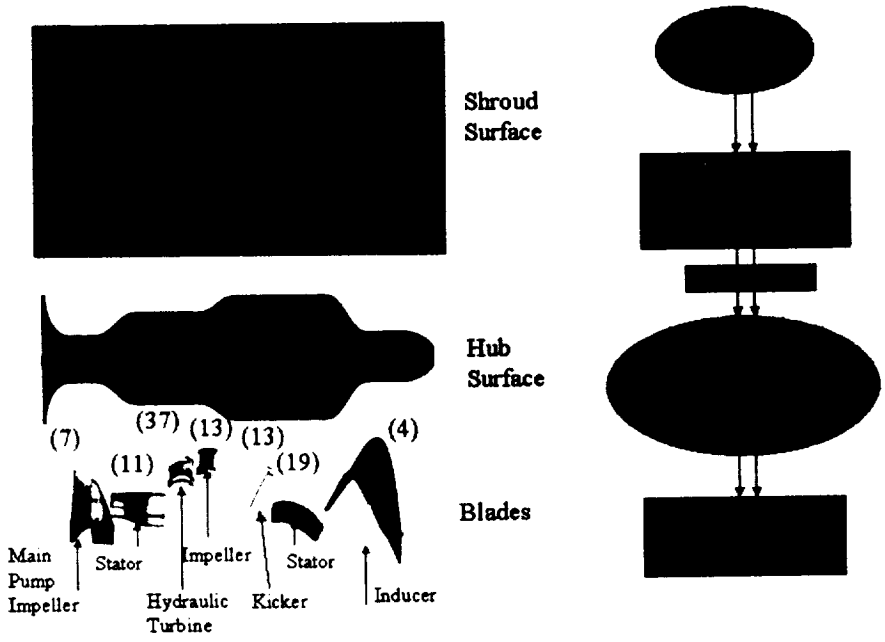


**Figure 3**. A pump model and steps taken in the simulation procedure.

In order to compute the flow on grids with moving boundaries, the overset grid scheme in OVERFLOW-D[7] code is incorporated with the INS3D solver that new connectivity data is obtained at each time step. The overlapped grid scheme allows sub-domains move relative to each other, and provides a general flexibility when the boundary movement creates large displacements. Figure 3 shows the model for boost pump and the steps taken in the simulation procedure. The numbers in figure 3 indicate the number of the blades in each section. The computational grid has been generated by using the OVERGRID[8] software. OVERGRID is a unified graphical interface for performing overset grid generation. The software contains general grid manipulation capabilities as well as modules that are specifically targeted for efficient creation of overlapping grids. General grid utilities include functions for grid transformation, redistribution, smoothing, concatenation, extraction, extrapolation, projection, and many others. Functions especially useful for overset grids include feature curve extraction, hyperbolic and algebraic surface grid generation, hyperbolic volume grid generation, and Cartesian box grid generation. Visualization is achieved using OpenGL while widgets are constructed with Tcl/Tk. The software is portable between various platforms from UNIX workstations to personal computers.

In order to demonstrate the current unsteady solution capability, the SSME shuttle upgrade pump configuration has been selected. Figure 4 shows the geometry of the test rig for this pump being tested at NASA-MSFC facilities. In this particular configuration, the SSME impeller is unshrouded.

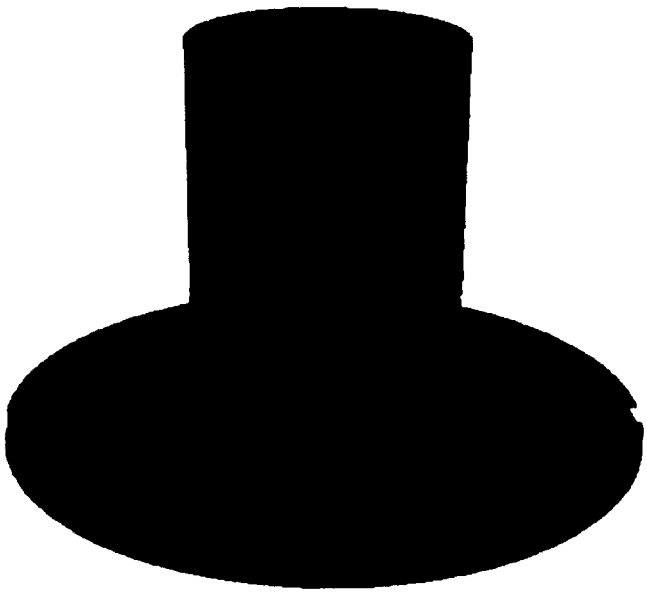

**Figure 4**. Geometry of SSME-rig1 shuttle upgrade pump impeller

The computational grid for the inlet guide vanes, impeller and diffuser sections of the SSME-rig1 configuration are shown in figures 5,6 and 7, respectively.

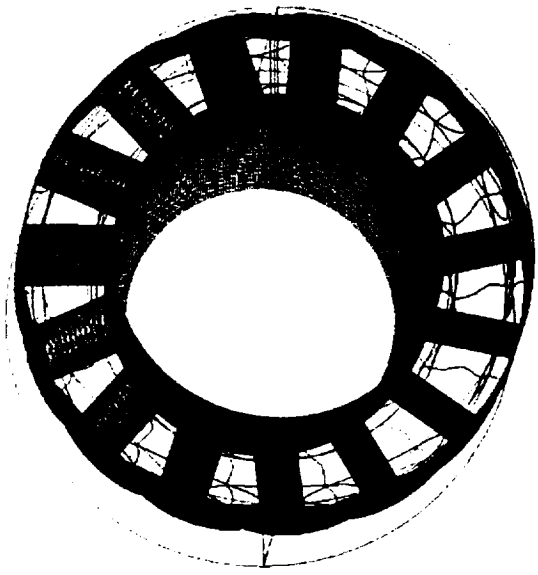

**Figure 5**. Computational grid of SSME-rig1 inlet guide vanes with 17 zones, and 5.5 Million grid points.

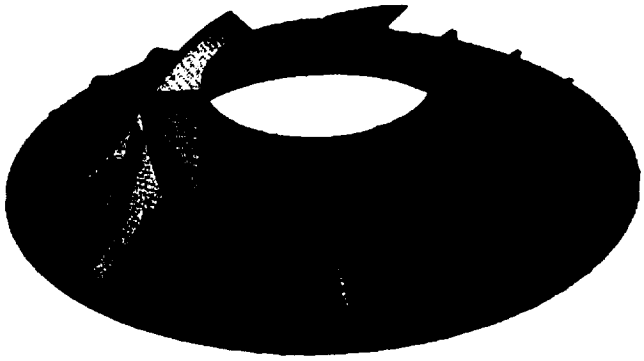

**Figure 6**. Computational grid of SSME impeller with 60 zones, and 19.2 Million grid points.

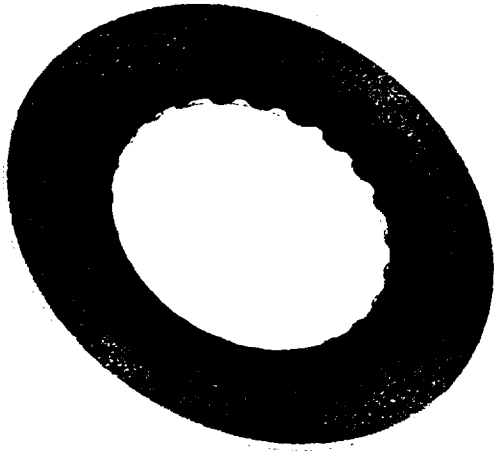

**Figure 7**. Computational grid of SSME-rig1 diffuser with 24 zones, and 6.5 Million grid points.

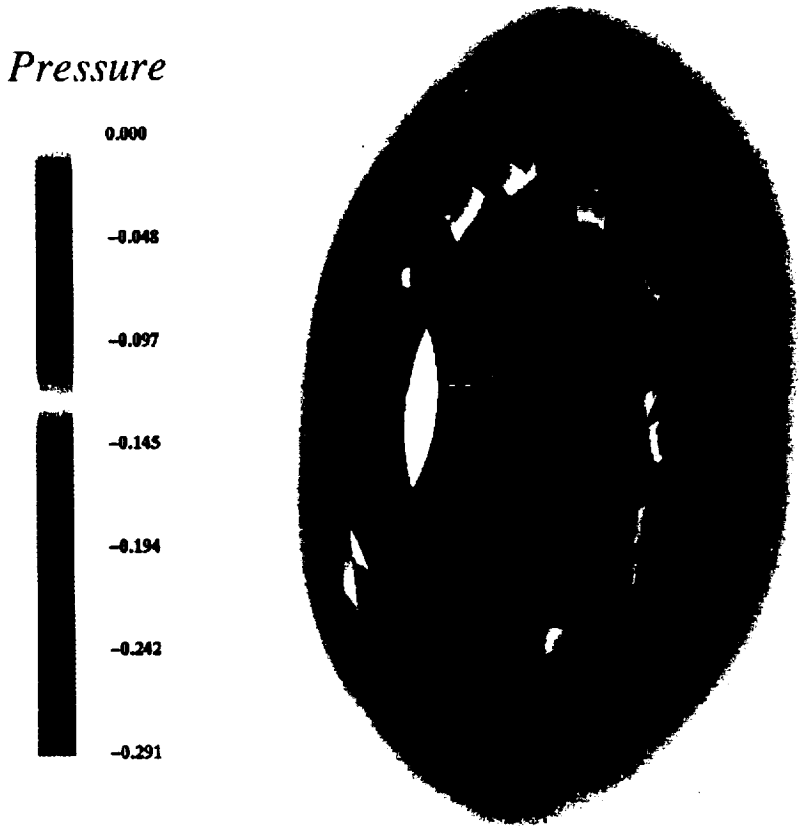


**Figure 8**. Computed surface pressure for SSME-HPFT impeller.

## COMPUTED RESULTS

Computed results are obtained for 2.8 Million and 19.2 Million grid points SSME impeller models. Figure 8 shows computed surface pressure of the shrouded SSME impeller. The performance of the two approaches in obtaining multi-level parallelism of the INS3D code is reported in this section. The first approach is hybrid MPI/OpenMP and the second approach is Multi Level Parallelism (MLP) developed at NASA-Ames Research Center. The first approach is obtained by using massage-passing interface (MPI) for inter-zone parallelism, and by using OpenMP directives for intra-zone parallelism. INS3D-MPI[9] is based on the explicit massage-passing interface across MPI groups and is designed for coarse grain parallelism. The primary strategy is to distribute the zones across a set of processors. During the iteration, all the processors would exchange boundary condition data between processors whose zones shared interfaces with zones on other processors. A simple master-worker architecture was selected because it is relatively simple to implement and it is a common architecture for parallel CFD applications. All I/O was performed by master MPI process and data was distributed to the workers. After the initialization phase is complete, the program begins its main iteration loop.

The SSME impeller model with 24 zones, and total grid points of 2.8 Million is used as a test case for the coarse grain INS3D-MPI code. For this version of the code, the number of zones in the computational model limits the maximum number of CPU count. Figure 9 shows floating point counts per second for this computation on SGI Origin 2000 platform. The average speedup, as compared to the linear speedup, is about 65% for 24 processors. It should be noted that the number of MPI groups reported in this paper always include the master MPI process. For the first four processors, very good performance is obtained. When the number of CPU count is increased further, the performance of the code is decreased due to the load balancing issues. In order to obtain fine grain parallelism, OpenMP directives are utilized [10]. Figure 10 shows time (in seconds) required per time integration step versus number of processors from the hybrid parallel code. It should be noted that the "time per iteration" reported in this paper includes the time obtaining Ax=b linear system of equations and the time solving this particular system of equations for the entire grid system. In other words, the iteration term is used for the physical time step, not for the iteration of linear equation solver. It also should be noted that the number of implicit line relaxation sweeps is kept same at each time step. The cases for 4, 12, and 24 MPI groups were plotted in figure 10. For each MPI group, various numbers of threads, such as 1, 2, 4, 8, and 16, were used. The number of CPU count is equal the number of threads multiplied by the number of MPI groups. When number of threads is increased, the performance of the code slows down because the grid size for each zone is relatively small for higher number of threads. This is shown in figure 12.

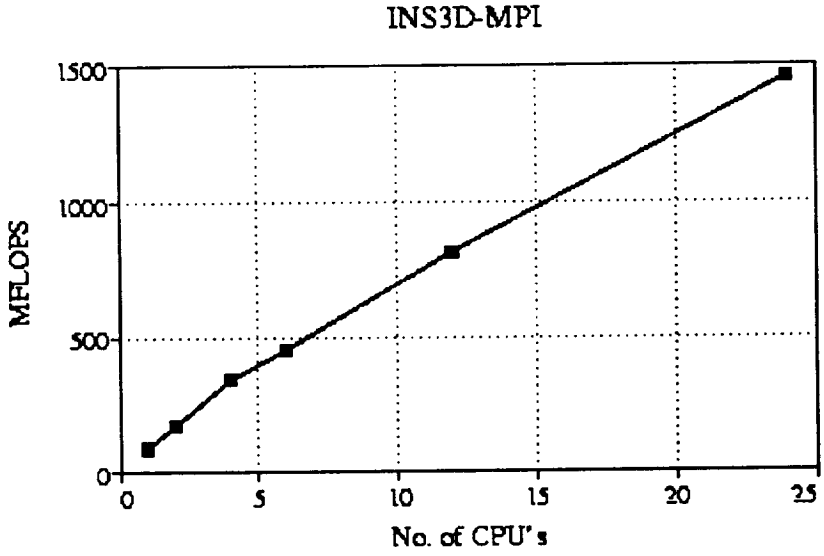


**Figure 9.** INS3D-MPI performance on the SGI O2K, SSME-HPFT Impeller.

When the problem size is increased from 2. 8 Million grid points to 19.2 Million grid points, the SSME impeller has 60 zones where the smallest zone has 74K grid points and the largest zone has 996K grid points. Figure 12 shows the effect of MPI groups on the performance of the code when one OpenMP thread is used. A good load balancing is obtained up to 20 MPI groups. When more than 20 MPI groups are employed, no more improvements in the performance of the

code is observed. The number of OpenMP threads is increased to 2, 4, 6, 10, and 15 for the same MPI groups. These cases are plotted in figure 13 and figure 15. Figure 14 shows time per iteration versus total CPU count, and figure 14 shows the cases for various OpenMP threads. The best performance was obtained for 20 MPI groups. In figure 13, the effect of load balancing can be seen for 30 MPI groups. The OpenMP directives show very similar speed-up for 20 and 30 MPI groups (see figure 14).

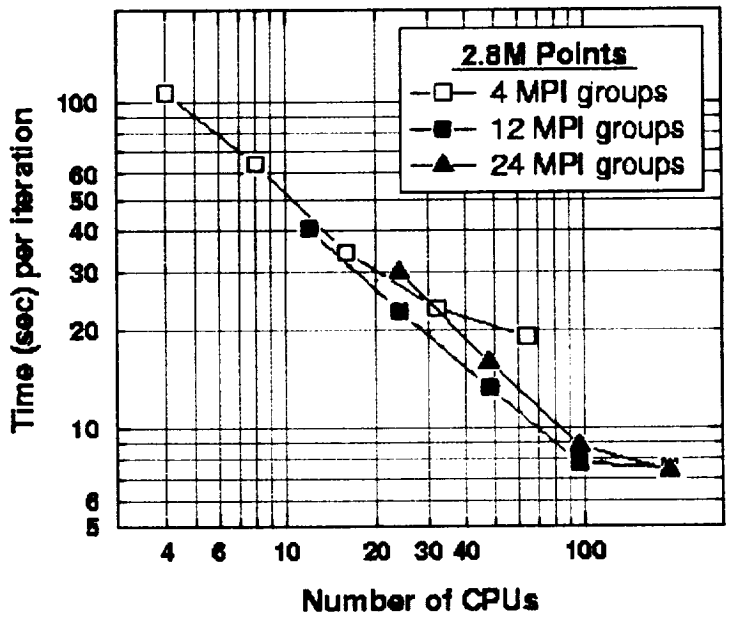


**Figure 10.** Time per iteration step for various MPI groups (INS3D MPI/OpenMP).

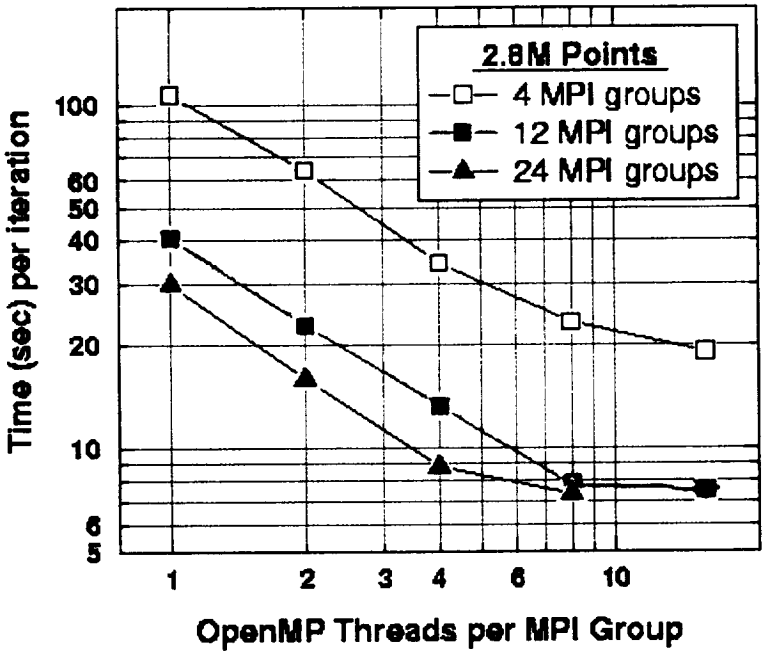


**Figure 11.** Time per iteration step versus OpenMP threads per MPI group.

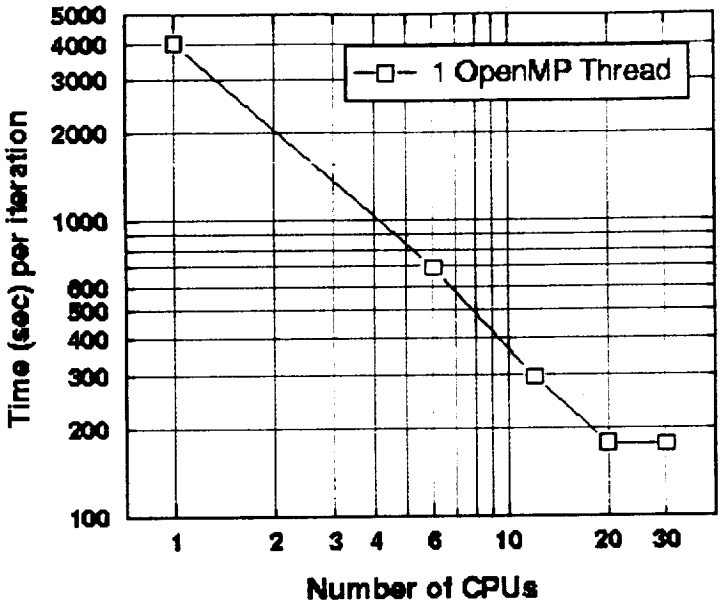


**Figure 12.** Performance for various MPI groups when one OpenMP thread is used (SSME impeller 19.2 Million grid).

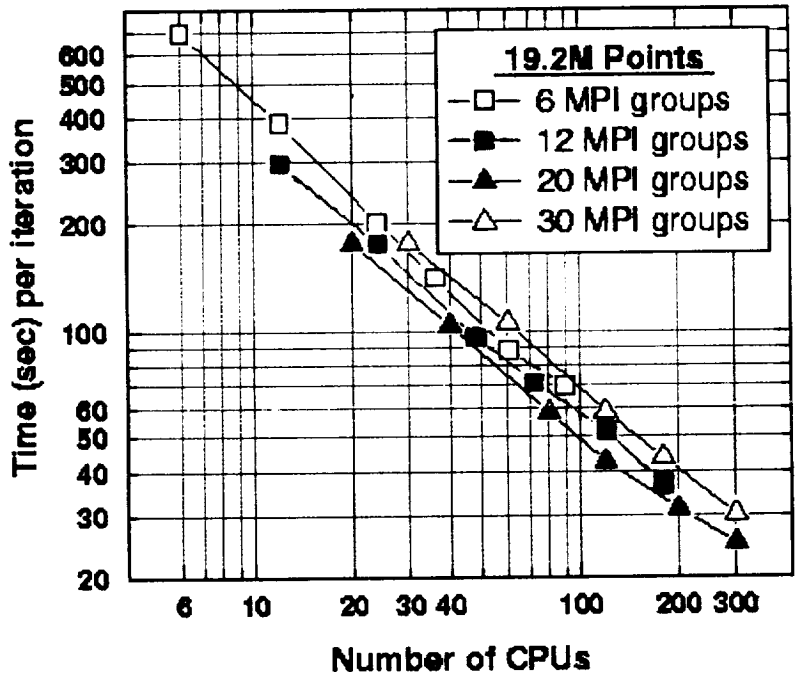


**Figure 13.** INS3D-MPI/OpenMP performance versus CPU counts for Origin 2000.

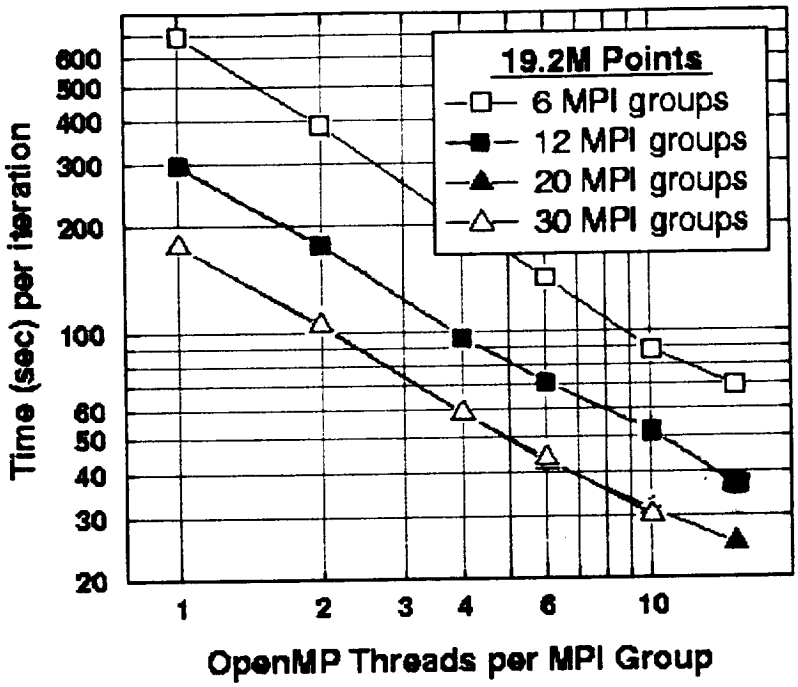


**Figure 14.** INS3D-MPI/OpenMP performance versus OpenMP threads for various MPI groups.

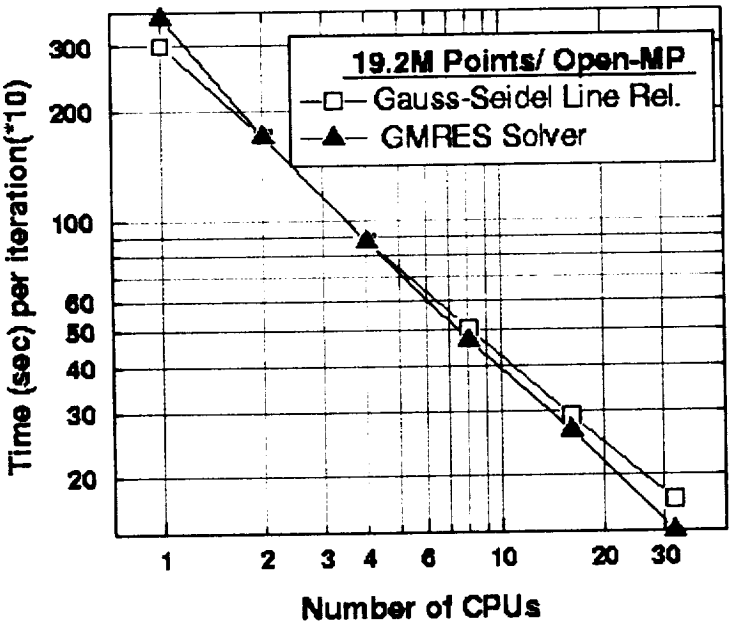


**Figure 15.** OpenMP performance for two different linear solvers.

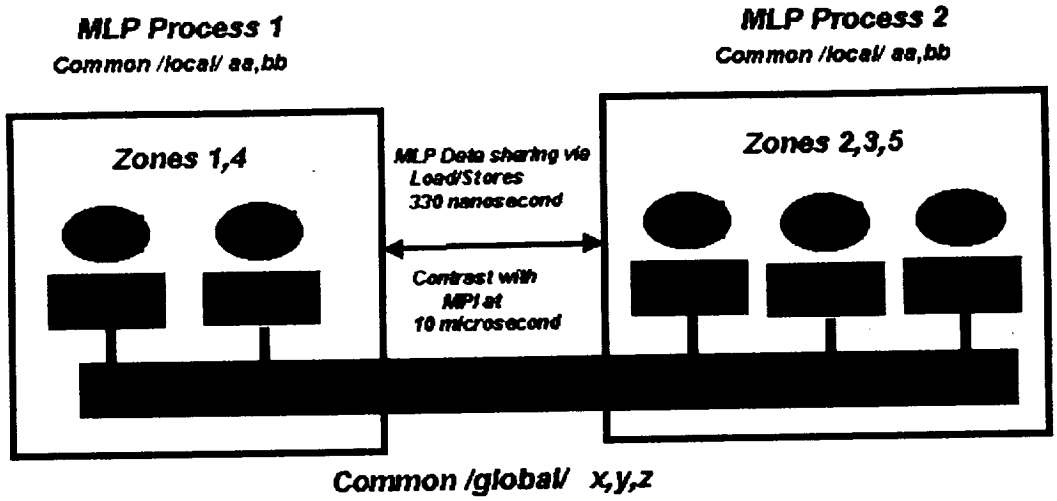


**Figure 16.** Shared memory MLP organization for NAS-MLP [11].

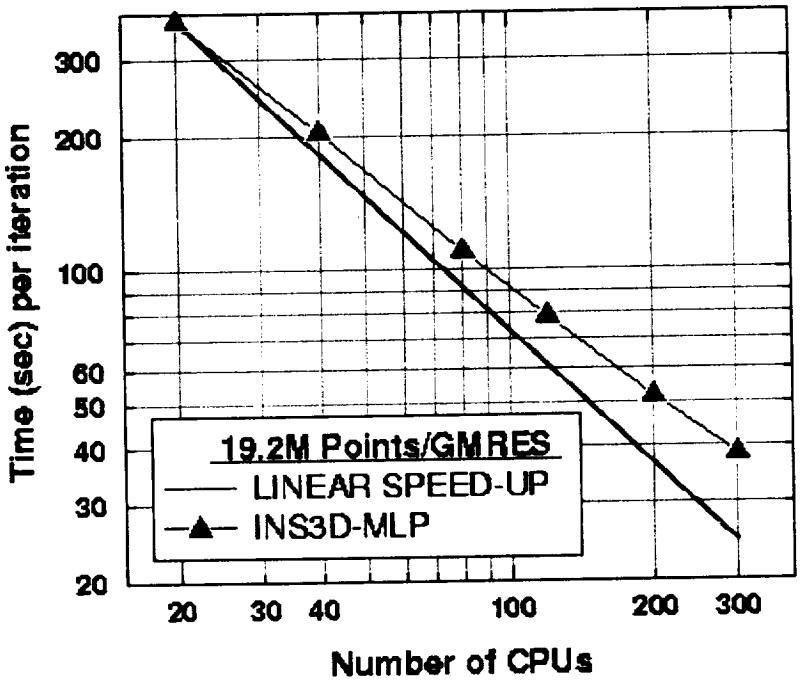


**Figure 17.** INS3D-MLP performance versus CPU counts for Origin 2000.

Figure 15 shows the effect of OpenMP threads for implicit Gauss-Seidel line-relaxation scheme and GMRES-ILU(0) scheme. Four Gauss-Seidel sweeps were performed for line-relaxation scheme, and 20 directional searches were performed for GMRES solver. The OpenMP directives show better speed-up for GMRES solver due to nested loop effect in oblique planes.

The second approach in Multi-Level Parallelism (MLP) is obtained by using NAS-MLP[11] routines developed by Taft for the OVERFLOW code. The shared memory

MLP technique developed at NASA Ames Research Center has been incorporated into the INS3D code. This approach differs from the MPI/OpenMP approach in a fundamental way in that it does not use messaging at all. All data communication at the coarsest and finest level is accomplished via direct memory referencing instructions. This approach also provides a simpler mechanism for converting legacy code, such as INS3D, then MPI. For shared memory MLP, the coarsest level parallelism is supplied by spawning of independent processes via the standard UNIX fork. The advantage of the UNIX fork over MPI procedure is that the user does not have to change the initialization section of the large production code. Shared memory MLP is inserted into INS3D in a very similar way that Taft inserts MLP into OVERFLOW code. Library of routines are used to initiate forks, to establish shared memory arenas, and to provide synchronization primitives. The shared memory organization for INS3D is shown in figure 16. The boundary data for the overset grid system is archived in the shared memory arena by each process. Other processes access the data from the arena as needed. Figure 17 shows INS3D-MLP performance versus CPU count for 19.2 Million-grid points SSME impeller model. GMRES-ILU(0) linear solver was used for these computations. The MLP version of code shows 73% of the linear speed-up performance. When MLP performance is compared with MPI/OpenMP performance (figure 13, 20 MPI groups), 19% more speed up is observed by using MLP version of the code. This comparison can be seen in figure 18. It should be noted that this comparison is preliminary since the further improvements in the fine-grain parallelization of the MLP code are currently underway.

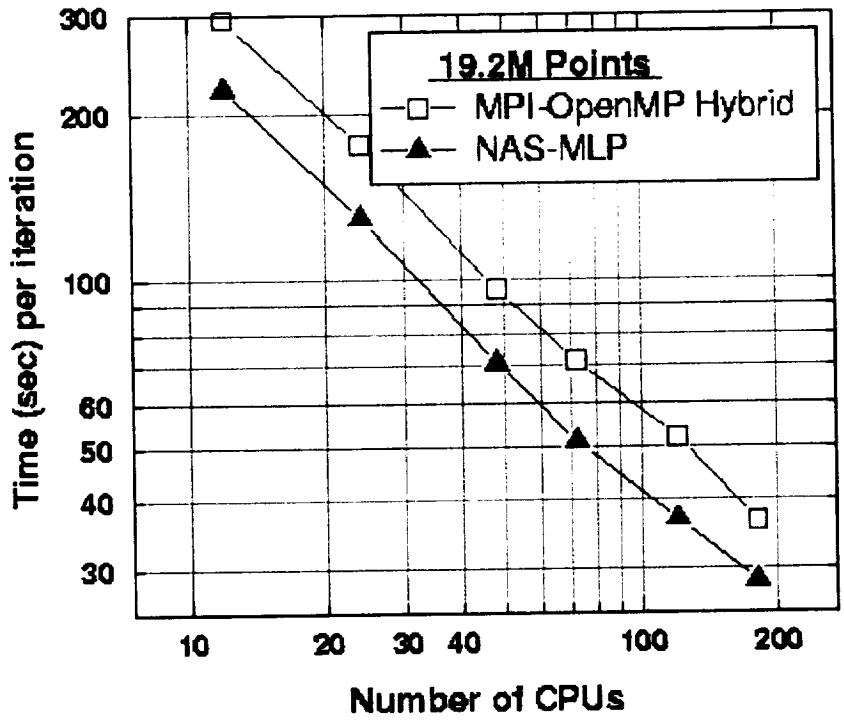


**Figure 18.** Comparison of INS3D-MPI/OpenMP and INS3D-MLP performance.

## SUMMARY

An incompressible flow solver in steady and time-accurate formulations has been utilized for parallel turbo-pump computations. Grid systems and numerical procedures are outlined for unsteady turbo pump simulations. Results from 2.8 Million and 19.2 Million grid points SSME impeller models are presented for the performance evaluations of the INS3D-MPI/OpenMP and INS3D-MLP versions of the code. SSME impeller model with 60 zones showed that up to 20 MPI groups hybrid code showed good scalability. OpenMP directives were more effective for GMRES(ILU) solver than line-relaxation scheme. Shared memory MLP version of the code was developed by using NAS-MLP routines. The SSME impeller computations showed very good scalability for the MLP version.

## ACKNOWLEDGEMENTS

Authors are grateful to Tom Faulkner and Jennifer Dacles Mariani for providing the coarse grain INS3D-MPI version of the code, to Henry Jin for his support and valuable ideas and discussions in OpenMP directives, and to James R. Taft for providing NAS-MLP routines.

## REFERENCES

1. Chorin, A., J., ''A Numerical Method for Solving Incompressible Viscous Flow Problems" ***Journal of Computational Physics***, Vol. 2, pp. 12-26, 1967.
2. Kwak, D., Chang, J. L C., Shanks, S. P., and Chakravarthy, S., "A Three-Dimensional Incompressible Navier-Stokes Flow Solver Using Primitive Variables," ***AIAA Journal*** , Vol. 24, No. 3, pp. 390-396, 1977.
3. Rogers, S. E., Kwak, D. and Kiris, C., "Numerical Solution of the Incompressible Navier-Stokes Equations for Steady and Time-Dependent Problems," ***AIAA Journal***, Vol. 29, No. 4, pp. 603-610, 1991.
4. Roe, P.L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," ***J. of Comp. Phys.***, Vol. 43, pp. 357-372 1981.
5. MacCormack, R., W., "Current Status of Numerical Solutions of the Navier-Stokes Equations," AIAA Paper No. 85-0032, 1985.
6. Rogers, S. E., "A Comparison of Implicit Schemes for the Incompressible Navier-Stokes Equations and Artificial Compressibility," ***AIAA Journal*** , Vol. 33, No. 10, Oct. 1995.
7. Meakin, Robert L., "Composite Overset Structured Grids," **Handbook of Grid Generation**, CRC Press, Eds. Thompson, Soni, Weatherill, 1998.
8. Chan, W. M., OVERGRID -- A Unified Overset Grid Generation Graphical Interface, to appear in ***Journal of Grid Generation***, 2000.
9. Faulkner, T., and Mariani, J., ``MPI Parallelization of the Incompressible Navier-Stokes Solver (INS3D). http://www.nas.nasa.gov/~faulkner/home.html
10. H. Jin, M. Frumkin and J. Yan, "Automatic Generation of OpenMP Directives and Its Application to Computational Fluid Dynamics Codes," in the

Proceeding of the Third International Symposium on High Performance Computing, Tokyo, Japan, Oct. 16-18, 2000.

11. Taft, J. R., Performance of the OVERFLOW-MLP and LAURA-MLP CFD Codes and the NASA Ames 512 CPU Origin Systems," HPCC/CAS 2000 Workshop, NASA Ames Research Center, 2000.